Reverse Hillclimbing, Genetic Algorithms and the Busy Beaver Problem

Terry Jones
Gregory J. E. Rawlins

SFI WORKING PAPER: 1993-04-024

SANTA FE INSTITUTE

Reverse Hillclimbing, Genetic Algorithms and the Busy Beaver Problem

Terry Jones * Gregory J. E. Rawlins †

April 29, 1993

Abstract

This paper introduces a new analysis tool called *reverse hillclimbing*, and demonstrates how it can be used to evaluate the performance of a genetic algorithm. Using reverse hillclimbing, one can calculate the exact probability that hillclimbing will attain some point in a landscape. From this, the expected number of evaluations before the point is found by hillclimbing can be calculated. This figure can be compared to the average number of evaluations done by a genetic algorithm.

This procedure is illustrated using the *Busy Beaver problem,* an interesting problem of theoretical importance in its own right. At first sight, a genetic algorithm appears to perform very well on this landscape, after examining only a vanishingly small proportion of the space. Closer examination reveals that the number of evaluations it performs to discover an optimal solution compares poorly with even the simplest form of hillclimbing.

Finally, several other uses for reverse hillclimbing are discussed.

*Santa Fe Institute, 1660 Old Pecos Trail, Suite A., Santa Fe, NM 87501, USA email: terry@santafe.edu

†Department of Computer Science, Indiana University, 215 Lindley Hall, Bloomington, IN 47405, USA. email: rawlins@cs.indiana.edu

1 Introduction

1.1 Fitness Landscapes

Many problems can be viewed as a search in a space of solutions. If the structure of a solution can be expressed as settings for each of n variables, one can imagine an n dimensional space with an axis for each of the variables. The *neighborhood* of a point P in the space is the set of all points that differ from P in exactly one variable. When each variable is binary, each point has n neighbors. These will be called P's *one–mutant* neighbors.

If the quality of a proposed solution can be quantified, then each point in the space has associated with it a number corresponding to its worth as a solution, called its *fitness.* The entire space, including the fitness values, is called a *fitness landscape* or simply a *landscape.* In this paper, higher quality solutions are assumed to have numerically higher fitness values. Initially, the fitnesses of all points in the space are unknown. The task is to identify high points on the landscape, or, ideally, the highest point(s).

Calculating the fitness of a point is called *evaluating* the point and is done by an *evaluation function.* A search algorithm is judged by the number of points it evaluates as well as the quality of the solutions it finds. Exhaustive search will always find the optimal solution, but the space is usually too large to make this an acceptable search method. At the other end of the spectrum, randomly choosing a single point requires just one evaluation, but quality is unlikely to be high. Between these extremes are many alternatives, including random search, genetic algorithms and the many forms of hillclimbing (such as gradient descent and simulated annealing).

Different problems have different landscapes, so an algorithm may be good on one type of landscape but not on another. Further, an algorithm may perform well in one region of a space and not in another. In addition, the probabilistic nature of most of these algorithms, makes it hard to predict what they will do from run to run. All of these issues make it hard to compare the performance of different algorithms. It is worthwhile to study smaller, better understood spaces and the performance of algorithms on them. There are many questions that we can answer, to some extent, on these landscapes: questions about specific algorithms, comparisons between algorithms, and questions about the nature of the landscapes themselves.

1.2 Genetic Algorithms

Genetic Algorithms, (GAs) were first proposed by Holland [7]. A GA maintains a population of points on the landscape and uses various methods modeled on genetics to evolve new, and hopefully fitter, populations. The ideas that are most commonly borrowed from genetics are mutation and recombination. Each solution is viewed as the genome of an individual in the population (see [6] for further references).

1.3 Hillclimbing and Reverse Hillclimbing

Hillclimbing typically chooses a random starting point in the landscape then attempts to move uphill to better one–mutant neighbors, until it can no longer do so. In *Steepest Ascent Hillclimbing* all the one–mutant neighbors of the current point are examined. If there are fitter neighbors, move to the fittest. If there are several equally fit best neighbors, choose one at random.

To properly evaluate a hillclimber both in relation to other hillclimbers and in relation to GAs it is important to answer the question: What is the probability that a hillclimb, started from a random point in the space, will discover a given point P? This question was formerly an extremely difficult one to answer on a rugged landscape. Reverse hillclimbing answers it very quickly and exactly. In fact, the more rugged and difficult to search a landscape is, the better is the performance of reverse hillclimbing. Formerly, it was at worst computationally infeasible and at best slow and inaccurate to even approximate this probability via repeated random hillclimbs.

The question can be answered by locating all the points in P's basin of attraction and the probability that each will hillclimb to P. These probabilities can then be summed and the total divided by the size of the space. Reverse hillclmibing does this by descending from P in all possible directions, but only to points that could possibly re-ascend to P.

The algorithm has at heart a recursive function **reverse_hillclimb()** that takes as arguments a point P and a probability p that the point hillclimbs to P. Initially, the function is called with some peak P and $p = 1$. A global hash table keeps track of all the points that have been encountered, their probabilities of climbing to P, and a pointer to a list of their interesting descendants (see the second and third items below). The function does the following:

1. If P is not in the hash table, add it and its probability p, otherwise, increment the existing probability by p.

2. Make a list L of all P's one–mutant neighbors that are less fit than P. This can be obtained from the hash table if P has been seen before.

3. For each element Q of L, check to see if Q hillclimbs back to P, and if so, determine the probability, q that it does so. Recurse, calling **reverse_hillclimb**(Q, pq).

When the recursion is finished, it is simple to calculate the probability that a single steepest ascent hillclimb from a randomly chosen point in the space will reach P.

The following two questions can now be answered: Given that a GA has located a certain peak, how long, on average, would it take a hillclimbing algorithm to find the same peak, and how does this compare with the time the GA takes?

The reciprocal of the probability that reverse hillclimbing produces is the expected number of hillclimbs it will take until P is encountered. Using this number and an estimate, obtained through sampling, of the average uphill walk length, one obtains the expected number of evaluations before P is found. This number can be compared directly with the

k	$\Sigma(k)$	$S(k)$
1	1	1
2	4	6
3	6	21
4	13	107
5	$\geq$ 4,098	$\geq$ 47,176,870
6	$\geq$ 95,524,079	$\geq$ 8,690,333,381,690,952

Table 1: The State of The Busy Beaver World

observed performance of the GA, thus giving information that can be used to better tune the GA or to determine whether the GA is doing useful work.

1.4 The Busy Beaver Problem

In the Busy Beaver problem [13] deterministic Turing Machines (TMs) with alphabet {0, 1} and a fixed number of states k (plus a halt state) are sought that write as many 1 symbols as possible on an initially zero–filled two–way infinite tape before halting. The number of 1 symbols that remain on the tape is the machine's score. The problem is to find, for a given k, the machine with the highest score, a number denoted by $\Sigma(k)$. The number of steps the machine makes before halting is called the *shift number* and is denoted $S(k)$. More than one machine may generate $\Sigma(k)$ ones and these machines do not necessarily have the highest shift numbers amongst all k–state machines.

The problem is intractable. The function defined by $\Sigma(k)$ is not computable, it grows faster than any computable function. Even $k = 2$ is difficult to do by hand. Thirty years ago Rado [12] wrote: "Even though skilled mathematicians and experienced programmers attempted to evaluate $\Sigma(3)$ and $S(3)$, there is no evidence that any known approach will yield the answer, even if we avail ourselves of high–speed computers and elaborate programs. As regards $\Sigma(4)$, $S(4)$ the situation seems to be entirely hopeless at present."

Today we know that his pessimism was justified; exact answers are known only for $k \leq 4$. These proofs have been developed as much through concerted human effort as through brute-force computer enumeration [2, 3]. See [3] and [9] for details of $k = 4$ and 5 respectively. The figures for $k = 6$ in Table 1 have not yet been published or independently verified [10].

As further illustration of the growth of $\Sigma(k)$, Schult (see [5]) found a 12–state machine that generates

$$6 * 4096^{4096^{\cdots 4096^4}}$$

ones. The ellipsis in the exponent stack represents 163 missing exponents, each of 4,096.

The number of possible machines on k states is, in the simplest enumeration, $(4 + 4k)^{2k}$. In the range of interest, $3 \leq k \leq 9$, this number increases by three orders of magnitude for each increment of k. Further, the difficulty of the problem is greatly compounded by

the extreme growth in the shift number. From Table 1 it can be seen that the shift number increases by a factor of over 400,000 as k increases from 4 to 5, and by over 180,000,000 when k moves from 5 to 6. This is important since testing machines may require simulating them for this many more steps. Without further theoretical insight, to discover a new 5–state champion, requires simulating machines for at least fifty million steps. The connection with the halting problem should be apparent.

Search for Busy Beavers in the past has been in two directions: finding faster ways to simulate TMs, and finding smarter ways to prove that machines do not halt. The first has seen elaborate software contrivances and even the construction of special hardware for running TMs. The second approach involves either static analysis of the TM's transition rules or dynamic analysis of the TM's behavior. So far exhaustive search combined with fast simulation and good inferencing has been the source of all the best champions found.

2 Searching For Busy Beavers With A GA

2.1 Coding and Parameters

We used a character–based encoding. A TM with k states was represented by a character string of $6k$ bytes. When the machine is in a certain state, looking at a certain symbol, it needs to know three things: the next state, the symbol to write on the tape and the direction in which to move. Since the machine could be scanning either a 0 or a 1, six bytes are required. By restricting crossover to byte boundaries, only legal TM's are generated.

For the results reported below, the following GA parameters were used: tournament selection (of size 2) with the higher fitness individual selected with probability 0.75, mutation at a rate of 0.01 (a mutation always causes an allele change), two point crossover with probability 0.75, 50 generations, and a generation gap of 0.9 (the surviving 10% were the best from the generation). The population size varied with k. We used populations as large as 2,000 for $k = 4$. Approximately twenty different parameter settings were tried and the above settings did as well as any.

2.2 Results

In all the cases where the optimal score is known ($k \leq 4$), the GA found a machine that achieves it. We have not attempted runs for $k = 5$ due to the large number of simulation steps it is apparently necessary to run machines for in order to find a new champion. The results are summarized in Table 4.

For $k = 4$ we were encouraged to find that the GA discovered an optimal solution after examining, on average, only 0.7% of the entire space, and this with a naive TM simulator. But how did the GA find the optimal machine? Was it luck? Was this performance good? Disabling crossover produced a marked decrease in performance. Was crossover finding and using good schema? Attempting to answer these questions led to reverse hillclimbing.

k	Size of Space	$\Sigma(k)$	Number of Peaks
1	64	1	16
2	20,736	4	4
3	16,777,216	6	40
4	25,600,000,000	13	48

Table 2: The Number of Peaks of Optimal Fitness for $k \leq 4$

k	P(Locate)	Climbs		Walk Length		Neighbors	Evals
1	1.0	1	(0)	1.00	(0.71)	6	6
2	0.039	25	(25)	1.33	(0.85)	16	544
3	0.0045	220	(220)	1.61	(1.02)	30	10,650
4	0.0000020	497,129	(497,129)	1.87	(1.15)	48	44,741,100

Table 3: Expected Evaluations to Find Optimal TMs Using Steepest Ascent

3 What Reverse Hillclimbing Can Tell Us

With no more than the machines found by the GA in hand, we could have made progress with reverse hillclimbing. But, happily, the problem has a thirty year history and we were able to draw on the work of others who had found the peaks years earlier. It is important to note, however, that reverse hillclimbing *does not* need to first know the highest peaks, it can be used with any set of points and any space to derive information both about the space and about the probable performance of hillclimbing algorithms on that space. Thus, the following comparisons would be valid even if we only had information discovered through using only a GA.

The landscapes for $k \leq 4$ were examined in some detail. Once again, $k = 5$ is not considered here as nothing, apart from lower bounds, can be said about it with certainty. Table 2 shows the number of optimal peaks in these four landscapes.

Table 3 shows the results of reverse hillclimbing from these peaks. The second column shows the probability that a single hillclimb, started from a randomly chosen point in the space, will reach an optimal peak. This figure is calculated via reverse hillclimbing, and is exact (in the decimal places shown). The third column shows the expected number of hillclimbs until a peak is found, with standard deviations in parentheses. This is just the reciprocal of the probability of a successful ascent, this being the expected value of a geometric random variable. The fourth column gives the average length of a steepest ascent climb on the landscape (found via sampling). Again, standard deviations are given in parentheses. The fifth column shows the number of neighbors that must be examined in each step of a steepest ascent. This is given by $2k(k+2)$. The final column gives the

k	GA Evals		Hillclimb Evals	
1	8	(9.27)	6.005	(4.245)
2	8,117	(8,663)	542	(508)
3	123,404	(128,022)	10,606	(10,492)
4	186,666,666	(89,449,934)	42,372,351	(19,123,501)

Table 4: The GA Versus Steepest Ascent Hillclimbing

expected number of evaluations until a peak is found – this is the product of columns three and four and five.

For the $k = 4$ case, reverse hillclimbing found the probability after evaluating only 161,748 points out of 25,600,000,000 (0.0006% of the space). This figure would be very hard to obtain with any accuracy via sampling with iterated hillclimbing, which would only arrive at an optimal peak approximately once in a million evaluations, and of course for the Busy Beaver problem, 'evaluation' is often not cheap.

We can now compare these expectations against the performance of the GA.

4 Comparing The GA With A Hillclimber

Table 4 shows the average number of evaluations taken by the GA and by a steepest ascent hillclimber. Standard deviations follow the means. Note the close correspondence between the hillclimbing results and the predicted results in Table 3.

The hillclimber is finding peaks for $k = 4$ about four and a half times as fast as the GA. This does not necessarily imply that *every* GA would be worse than a hillclimber on this problem. The main point here is that we can use reverse hillclimbing to quickly and accurately assess how easily a hillclimber would find a result that a GA found. This is easily compared to the GA's performance and from this comparison one may be content with the GA, modify GA parameters, or even change search algorithms.

5 Discussion

Reverse hillclimbing has several other potential applications. Firstly, some observations:

- Any hillclimber can be reversed if it satisfies two conditions. The first is that it must be possible to calculate the algorithm's next possible moves and their probabilities. Secondly, the algorithm cannot be one which moves to a point whose fitness is less than or equal to the current point. Hillclimbers which violate the second condition can potentially loop indefinitely, and must be analyzed mathematically.

  These conditions allow us to perform reverse hillclimbing on all of *Steepest Ascent*, *Next Ascent*, *Random Ascent*, *First Ascent*, *Random Bit Climbing* ([4]), and various forms

of mutational hillclimbing, but not on *Stochastic Hillclimbing* ([1]), *Bit Setting* ([14]), *Random Mutation Hillclimbing* ([11]) or any of the forms of simulated annealing.

- One can do a roughly equivalent thing in the uphill direction, a *probabilistic ascent*, and find all the peaks that could be ascended to from P, each with a probability.

- There is a trade–off between how useful reverse hillclimbing will be and how rugged the landscape is. In the cases where the landscape is not rugged, hillclimbing can answer some questions quickly that reverse hillclimbing cannot, and vice–versa. If a peak has a small basin of attraction, reverse hillclimbing can be very useful.

Hillclimbing algorithms vary in terms of the amount of work done before a move is made and also in how deterministic they are. Less determinism generally means that a point can climb to more peaks, but each with smaller probability. Greater determinism means that peaks have smaller basins of attraction, but once a climb is started within one, the probability of reaching the peak is higher. These tradeoffs make it difficult to compare two hillclimbing algorithms both in general and for a specific landscape.

However with reverse hillclimbing, one can make direct comparisons between hillclimbing algorithms on a landscape. By sampling points, or using peaks found by hillclimbing or some other search, and reverse hillclimbing from them with different hillclimbers, one can tell which hillclimber will on average do better.

Using reverse hillclimbing and the probabilistic ascent mentioned above, one can obtain a measure of how a system in equilibrium, a spin glass for example, that is then disturbed is likely to settle, and how much energy is needed to bump the system into a new configuration. Reverse hillclimbing establishes the basin of attraction and probabilistic ascent then identifies the peaks that could be returned to and their probabilities.

If a landscape has peaks that are close together, as has been observed in NK landscapes [8], one could use reverse hillclimbing with probabilistic ascent to find the peaks that are close to a given peak. Reverse hillclimbing can establish the entire basin of attraction, and probabilistic ascent can then identify all peaks that can be ascended to from all these points. Any peak found using this method is 'close' to the original in the sense that there is at least one saddle point between them. Iterating this procedure could quickly identify small mountainous regions of the space. This is essentially a new search method that will work on spaces that have this characteristic.

The two methods can be used to arrive at statistical properties of a fitness landscape. For instance, the average number of points "below" and "above" a given point, or some measure relating the number of uphill directions to the number of peaks that can be reached.

Reverse hillclimbing could also be used in immune system modeling to determine the basin of attraction for a given antigen. One could then determine the set of existing antibodies that could climb to the antigen and their probabilities. Reverse hillclimbing could identify antigens that would not elicit an immune response, so–called "holes in the immune system" — this would happen if the model of the body had no antibody in the antigen's basin of attraction.

6 Conclusion

This paper introduced a new tool for landscape analysis, *reverse hillclimbing* and showed one way it can be used to examine the performance of a GA. The performance of a GA on the Busy Beaver problem was examined and further suggestions were made for the use of reverse hillclimbing and probabilistic ascent.

7 Acknowledgements

Many thanks to Allen Brady and Heiner Marxen who were very helpful and willing to share their knowledge and latest results on the Busy Beaver problem, despite the deluge of questions. Thanks too to Stuart Kauffman for his thoughts on the wider applications of reverse hillclimbing.

References

[1] Ackley, David H. [1987] *A Connectionist Machine For Genetic Hillclimbing*, Kluwer Academic Publishers.

[2] Brady, A. H. [1966], "The Conjectured Highest Scoring Machines for Rado's $\Sigma(k)$ for the Value $k = 4$" *IEEE Trans. on Electronic Computers* vol. EC–15, pp. 802.

[3] Brady, A. H. [1983], "The Determination of the Value of Rado's Noncomputable Function $\Sigma(k)$ for Four–State Turing Machines" *Mathematics of Computation*, vol. 40, no. 162. April 1983 pp. 647–665.

[4] Davis, Lawrence [1989] "Bit-Climbing, Representational Bias, and Test Suite Design" In Richard K. Belew and Lashon B. Booker (Eds.) *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 18–23.

[5] Dewdney, A. K. [1989] "Noncomputable Functions – The Busy Beaver Problem" *The Turing Omnibus*, Computer Science Press, Ch. 36, pp 241-244.

[6] Goldberg, David E. [1989], *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison–Wesley, 1989.

[7] Holland, John H. [1975], *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[8] Kauffman, Stuart A. [1989], "Adaptation on Rugged Fitness Landscapes" *Lectures in the Sciences of Complexity*, Ed. D. Stein, Addison–Wesley Longman, vol. 1, pp. 527–618.

[9] Marxen, H. and Buntrock, J. [1990], "Attacking The Busy Beaver 5" *Bulletin of The European Association for Theoretical Computer Science*, vol. 40, pp. 247–251.

[10] Marxen, H. [1992]. Personal communication.

[11] Palmer, Richard. [1992]. Personal communication.

[12] Rado, T. [1963], "On a Simple Source for Non–computable Functions." *Proceedings of the Symposium on Mathematical Theory of Automata*, Polytechnic Institute of Brooklyn, April 1963, pp. 75–81.

[13] Rado, T. [1962], "On Non-computable Functions" *Bell System Technical Journal*, vol. 41, pp. 877–884.

[14] Wilson, Stewart W. [1991] "GA–Easy Does Not Imply Steepest–Ascent Optimizable" In Richard K. Belew and Lashon B. Booker (Eds.) *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 85–89.